Naveen Narayanan Meyyappan (nm941) 12/19/2019

Fatima AlSaadeh (fya7)

Probability and Statistics

Predicting Diabetics in Pima Indians

In this project, we will design and develop a supervised learning model for predicting whether a patient has diabetes or not. Diabetes is a disease that occurs when blood glucose, also called blood sugar, is too high. Blood glucose is the main source of energy and comes from the food we eat. Insulin is a hormone made by the pancreas that allows our body to use sugar (glucose) from carbohydrates in the food that we eat for energy or to store glucose for future use. Insulin helps keeps our blood sugar level from getting too high (hyperglycemia) or too low (hypoglycemia). When our body does not produce enough insulin to convert the glucose, the excess glucose then stays in our blood and doesn't reach our cells. Over time, having too much glucose in our blood can cause health problems. Although diabetes has no cure, we can take steps to manage our diabetes and stay healthy. In this project, we will develop a machine learning model to predict whether a patient has diabetes or not based on different characteristics such as Blood Pressure, Skin Thickness, Insulin Level, BMI, Plasma Glucose Level and so on. The data set for this project is taken from the National Institute of Diabetes and Digestive and Kidney Diseases. All patients in this data set are females at least 21 years old of Pima Indian heritage. The Pima Indians (or Akimel O'odham, also spelled Akimel O'otham, "River People", formerly known as Pima) are a group of Native Americans living in an area consisting of what is now central and southern Arizona. Diabetes is an increasingly prevalent chronic disease characterized by the body's inability to metabolize glucose. Finding the disease at an early stage helps reduce medical costs and the risk of patients having more complicated health problems. This modeling project will help detect diabetes at early stages and will help us take effective action and to stay healthy.

Data Source

https://www.kaggle.com/uciml/pima-indians-diabetes-database This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases

Data Dictionary

Pregnancies: Number of times pregnant Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test

Blood Pressure: Diastolic blood pressure (mm Hg)

Skin Thickness: Triceps skin fold thickness (mm) Insulin: 2-Hour serum insulin (mu U/ml) BMI: Body mass index (weight in kg/(height in m)^2)

Diabetes Pedigree Function: A function which scores likelihood of diabetes based on family history Age: Age (years) Outcome: Class variable (0 or 1) 268 of 768 are 1, the others are 0

Data Summary

```
# Reading the data
data <- read.csv(file = "diabetes.csv",header = TRUE,sep = ",")
summary(data)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
nrow(data) # Number of rows in data
```

```
## [1] 768
```

```
ncol(data) # Number of columns in data
```

```
## [1] 9
```

There are 768 observations (rows) of 9 variables (columns). All the 9 columns are read as numerical and the range of the values in each columns are summarized as given above.

Exploratory Data Analysis

In the above result we find that the outcome column in data is read as a numerical column but actually the outcome column is a categorical column (consists of 0s and 1s only). We will convert it to a categorical variable.

```
data$Outcome=as.factor(data$Outcome)
dataforplot=data
dataforplot$Outcome = revalue(data$Outcome, c("0"="Not Diabetic",
"1"="Diabetic"))
```

Here the outcome variable 1 indicates that the patient is affected by diabetes and 0 indicates that the patient is not diabetic.

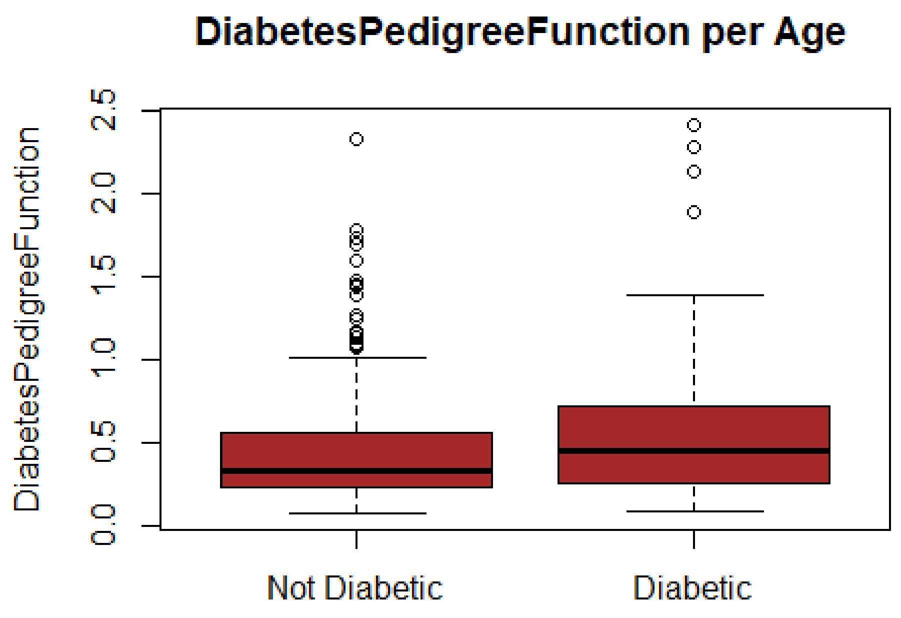
```
table(dataforplot$Outcome)

##
## Not Diabetic     Diabetic
##          500          268
```

Of the 768 patients considered about 200 of them are diabetic. Now let us plot some distributions to get some insights about the data.

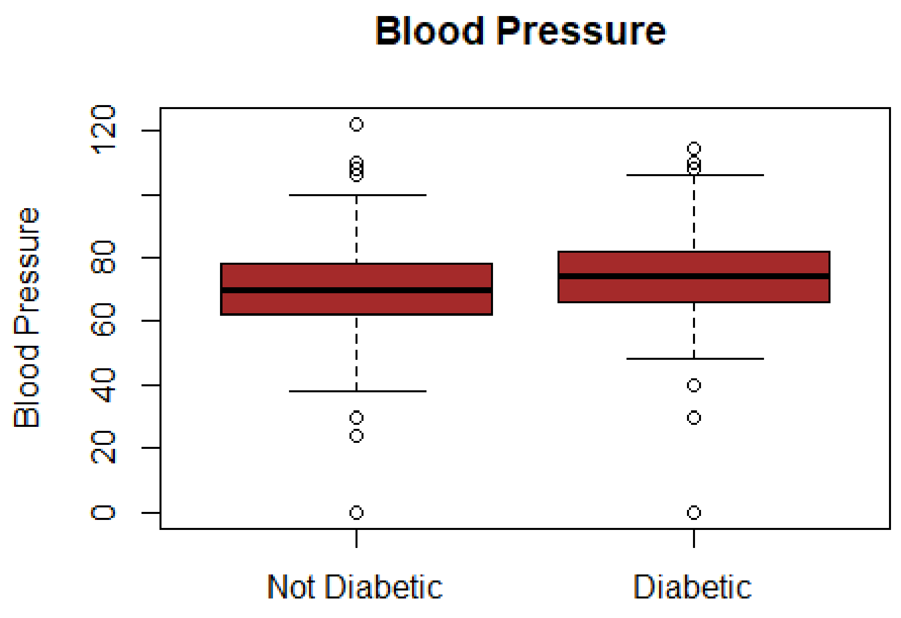
Diabetic Pedigree Function vs Outcome

```
boxplot(dataforplot$DiabetesPedigreeFunction~Outcome,data=dataforplot,
main="DiabetesPedigreeFunction per Age ",
        xlab="", ylab="DiabetesPedigreeFunction", col="brown")
```



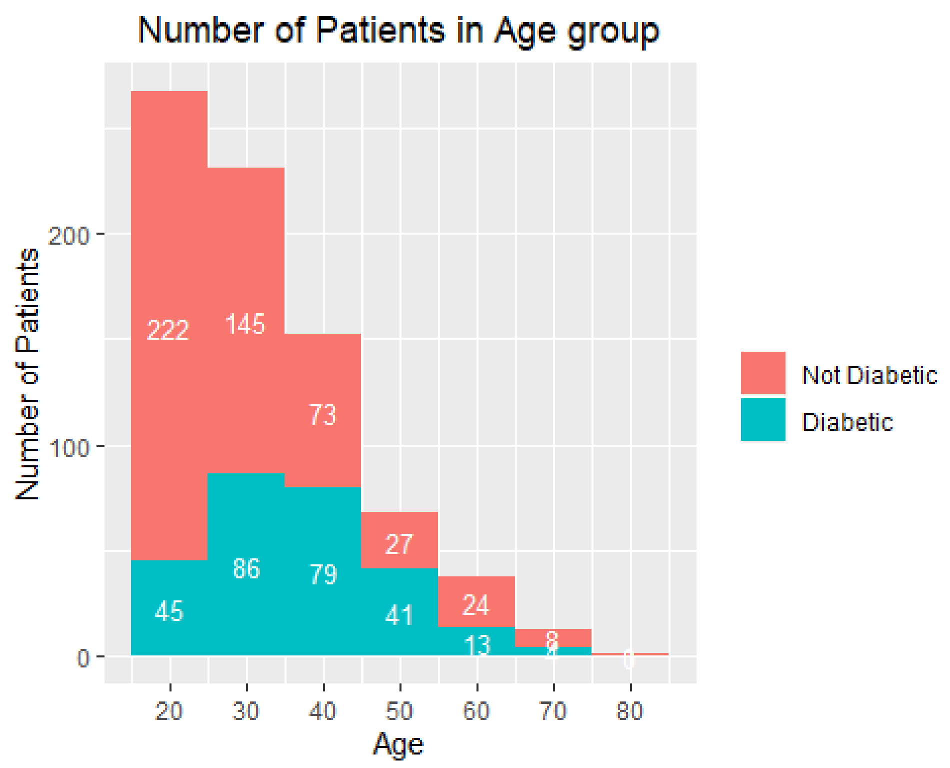
Blood Pressure vs Outcome

```
boxplot(dataforplot$BloodPressure~Outcome,data=dataforplot, main="Blood
Pressure ",
        xlab="", ylab="Blood Pressure", col="brown")
```



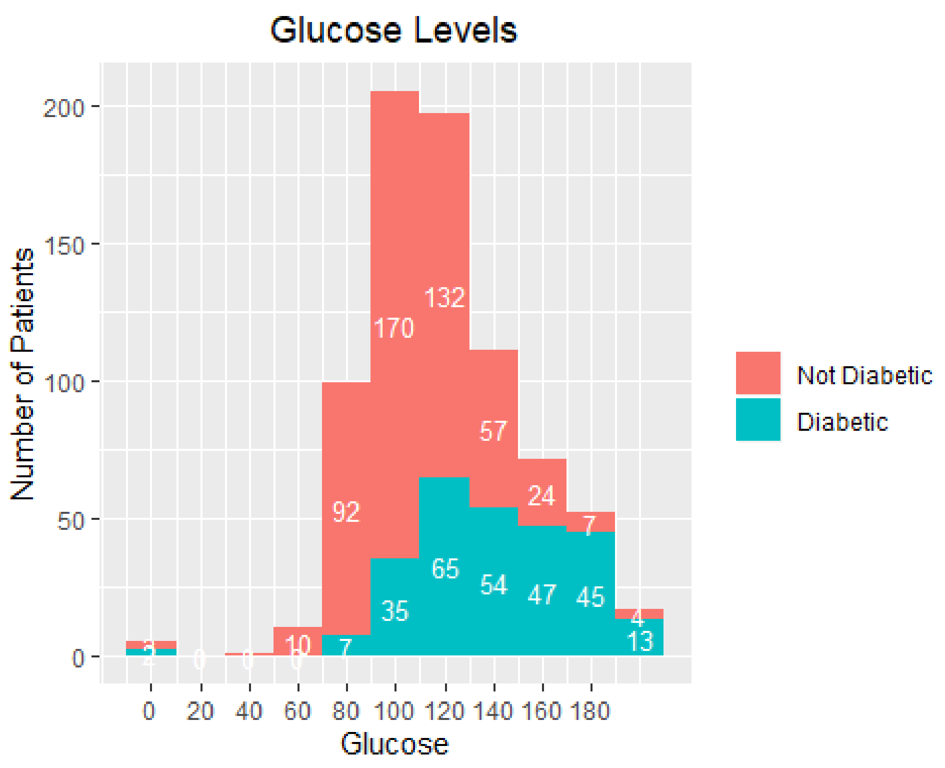
Age vs Outcome

```
ggplot(dataforplot, aes(fill=dataforplot$Outcome, x=dataforplot$Age)) +
        geom_histogram(position="stack", stat="bin", binwidth = 10) +
        stat_bin(binwidth=10, geom="text", colour="white", size=3.5,
aes(label=..count.., group=dataforplot$Outcome, y=(..count..)),
position=position_stack(vjust=0.5)) +
        scale_x_continuous(name="Age",breaks=seq(0,max(data$Age), 10)) +
        scale_y_continuous(name="Number of Patients") +
        ggtitle("Number of Patients in Age group") +
        scale_fill_discrete(name = "") +
        theme(plot.title = element_text(hjust = 0.5))
```



Glucose vs Outcome

```
ggplot(dataforplot, aes(fill=dataforplot$Outcome, x=dataforplot$Glucose)) +
        geom_histogram(position="stack", stat="bin", binwidth = 20) +
        stat_bin(binwidth=20, geom="text", colour="white", size=3.5,
aes(label=..count.., group=dataforplot$Outcome, y=(..count..)),
position=position_stack(vjust=0.5)) +
        scale_x_continuous(name="Glucose",breaks=seq(0,max(data$Glucose),
20)) +
        scale_y_continuous(name="Number of Patients") +
        ggtitle("Glucose Levels") +
        scale_fill_discrete(name = "") +
        theme(plot.title = element_text(hjust = 0.5))
```

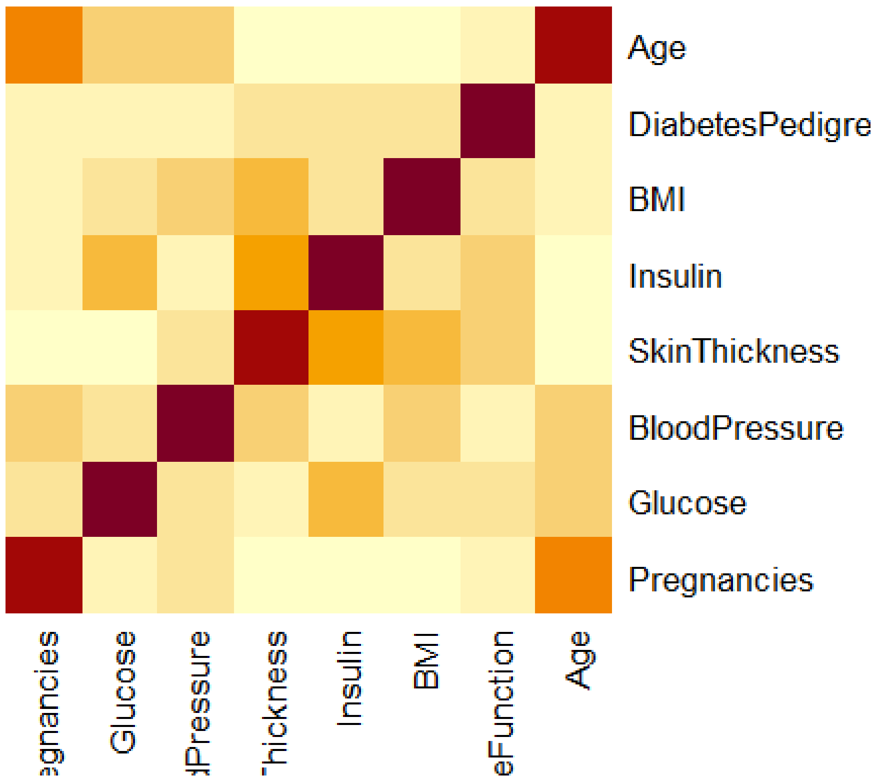


Correlation matrix and heatmap

```
cor(dataforplot[,1:8])

##                          Pregnancies    Glucose BloodPressure
## Pregnancies               1.00000000 0.12945867    0.14128198
## Glucose                   0.12945867 1.00000000    0.15258959
## BloodPressure             0.14128198 0.15258959    1.00000000
## SkinThickness            -0.08167177 0.05732789    0.20737054
## Insulin                  -0.07353461 0.33135711    0.08893338
## BMI                       0.01768309 0.22107107    0.28180529
## DiabetesPedigreeFunction -0.03352267 0.13733730    0.04126495
## Age                       0.54434123 0.26351432    0.23952795
##                          SkinThickness     Insulin        BMI
## Pregnancies                -0.08167177 -0.07353461 0.01768309
## Glucose                     0.05732789  0.33135711 0.22107107
## BloodPressure               0.20737054  0.08893338 0.28180529
## SkinThickness               1.00000000  0.43678257 0.39257320
## Insulin                     0.43678257  1.00000000 0.19785906
## BMI                         0.39257320  0.19785906 1.00000000
## DiabetesPedigreeFunction    0.18392757  0.18507093 0.14064695
## Age                        -0.11397026 -0.04216295 0.03624187
##                          DiabetesPedigreeFunction         Age
## Pregnancies                           -0.03352267  0.54434123
## Glucose                                0.13733730  0.26351432
## BloodPressure                          0.04126495  0.23952795
## SkinThickness                          0.18392757 -0.11397026
## Insulin                                   0.18507093 -0.04216295
## BMI                                       0.14064695  0.03624187
## DiabetesPedigreeFunction                  1.00000000  0.03356131
## Age                                       0.03356131  1.00000000

heatmap(cor(dataforplot[,1:8]),Colv = NA, Rowv = NA, scale="column")
```



The correlation matrix and the heat map shows that no two variables are having high correlation. The highest correlation is between Age and Pregnancies (0.54) which is not significant enough (>0.7) to be explored more.

Modelling

Now let us develop a model to predict if a patient has diabetes or not. For this the first step would be to split the data into train and test data sets.

```
# 2/3 of the data is used for training and 1/3 is used as testing data set
index_train<-sample(768,512)
train_set <- data[index_train, ]
test_set <- data[-index_train, ]
```

Logistic Regression Model

We will start off with Logistic Regression. Logistic regression also called a logit model, is used to model dichotomous outcome variables. In the logit model, the log odds of the outcome are modeled as a linear combination of the predictor variables. Logistic regression is conceptually similar to linear regression, where linear regression estimates the target variable. Instead of predicting values, as in the linear regression, logistic regression would estimate the odds of a certain event occurring. Logistic Regression is a supervised parametric learning model. In other words, the data should obey certain assumptions before developing the logistic regression model.

Assumptions of Logistic Regression Model

Assumptions of Logistic Regression Model 1. Binary logistic regression requires the dependent variable to be binary and ordinal logistic regression requires the dependent variable to be ordinal. -> Our dependent variable (Outcome) is a binary categorical variable.

2. Logistic regression requires observations to be independent of each other. In other words, the observations should not come from repeated measurements or matched data. -> Each row in the data set is independent of that of the other. Each row represents the data of a patient.

3. Logistic regression requires there to be little or no multi collinearity among the independent variables. This means that the independent variables should not be too highly correlated with each other. -> From the correlation matrix we found that there is not much collinearity between the columns of this data set.

4. Logistic regression assumes the linearity of independent variables and logs odds. although this analysis does not require the dependent and independent variables to be related linearly, it requires that the independent variables are linearly related to the log odds. -> The independent variables are linear with log-odds.

5. Logistic regression typically requires a large sample size. -> In our data set we have 798 observations which are large enough for 8 independent variables.

Building the Model

As all the assumptions are satisfied. Let us develop the model

```
train_set_logistic=train_set
test_set_logistic=test_set
logistic_model <- glm(Outcome ~ ., family = "binomial", data =
train_set_logistic)
print(summary(logistic_model))

##
## Call:
## glm(formula = Outcome ~ ., family = "binomial", data = train_set_logistic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5818  -0.7811  -0.4121   0.8129   2.7344
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.862506   0.867444  -9.064  < 2e-16 ***
## Pregnancies               0.135091   0.037055   3.646 0.000267 ***
## Glucose                   0.031188   0.004216   7.398 1.38e-13 ***
## BloodPressure            -0.015514   0.006209  -2.498 0.012474 *
## SkinThickness            -0.005805   0.007955  -0.730 0.465533
## Insulin                  -0.001451   0.001152  -1.259 0.208000
## BMI                       0.100082   0.018364   5.450 5.04e-08 ***
## DiabetesPedigreeFunction  1.007343   0.366837   2.746 0.006032 **
## Age                       0.012095   0.010758   1.124 0.260902
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.41  on 511  degrees of freedom
## Residual deviance: 509.35  on 503  degrees of freedom
## AIC: 527.35
##
## Number of Fisher Scoring iterations: 5
```

Understanding the Results

These were the results obtained for a logistic regression model. The result can be interpreted as follows:

Intercepts, Estimates, Standard Errors and P values.

We have the various columns and their estimate values. The estimate values indicate the slope for the best fit line and each column's value is multiplied by this slope. Then we have the standard error of the slope for each variable in the model. We also have the z scores associated with each column. The last column is the p value for these z scores. If the p value falls below 0.05(95% interval, then those columns are highlighted using the asterisk (*) symbol). The first row consists of the intercept which indicates the offset that has to be added to the model equation.

Null and Residual Deviance The null deviance shows how well the response variable is predicted by a model that includes only the intercept (grand mean) whereas residual deviance is the deviance with the inclusion of independent variables. Hence for calculating the null deviance, the degrees of freedom will be 512-1=511 and we have 503 degrees of freedom for residual deviance (503=512-8-1). We subtract 8 as we are having 8 independent variables. Residual is the difference between the actual and predicted value. So lower the residual score better the model.

AIC Akaike's Information Criterion (AIC) is -2log-likelihood+2k where k is the number of estimated parameters. It is useful for comparing models different models. Lower the AIC better is the performance of the model.

Fisher Scoring Iterations This is the number of iterations to fit the model. The logistic regression uses an iterative maximum likelihood algorithm to fit the data. The Fisher method is the same as fitting a model by iteratively re-weighting the least squares. It indicates the optimal number of iterations. Similar to Linear Regression, the model generates a linear equation using the given estimates and intercepts. Then the values from this equation will be converted into probabilities using the logit function. From the above result we find that Pregnancies, Glucose, Blood Pressure, Diabetes Pedigree Function, MI and Age are significant variables as they have the p values close to 0.05 or less than 0.05. The variable which does not contribute much for the model would be skin thickness.

Predicting in train and test data

Now let us try to predict the values in the train and test set using our developed model.

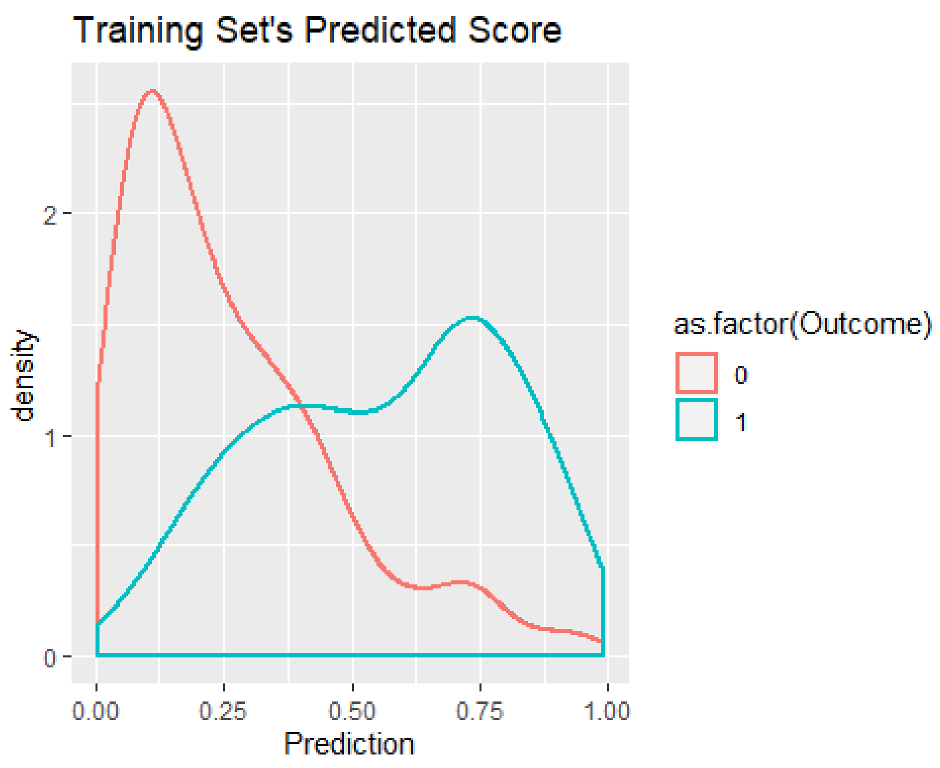
```
train_set_logistic$Prediction <- predict( logistic_model, newdata =
train_set_logistic, type = "response" )
test_set_logistic$Prediction  <- predict( logistic_model, newdata =
test_set_logistic , type = "response" )
```

The prediction column in both the data sets will now have probability values associated with each row. Now we need to convert this probability scores to whether a patient has diabetes or not using a threshold value.

Identifying the correct threshold value

Before setting the threshold let us take a look at the plot below:

```
# distribution of the prediction score grouped by known outcome
ggplot(train_set_logistic, aes(Prediction, color = as.factor(Outcome) ) ) +
        geom_density( size = 1 ) +
        ggtitle( "Training Set's Predicted Score" )
```



In the above plot We have two curves - red and blue. The red curve indicates the patients who do not have diabetes, and the blue curve shows the patient who has diabetes. The x-axis indicates the prediction probabilities, and the y-axis shows the number of data points. In an ideal situation, we would want the peaks of the two curves to be separated as much as possible. We also find that setting a threshold of 0.5 for the prediction score will not work out. From the graph, we find that the two curves cutoff at 0.3. The other reason why a threshold of 0.5 will not work out is that, in our sample out of the 768 samples, only 268 are affected by diabetes. This is not a 50-50 proportion. In other words, the probability of encountering a diabetic patient is less than that of a non-diabetic patient.

ROC Curve

So now, we need to find a proper threshold for our model. The (Receiver Operating Characteristics) ROC curve will now help us to fix the threshold. A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall, or probability of detection in machine learning. The false-positive rate is also known as the probability of false alarm and can be calculated as (1 − specificity). It can also be thought of as a plot of the power as a function of the Type I Error of the decision rule (when the performance is calculated from just a sample of the population, it can be thought of as estimators of these quantities). So let us plot our ROC curve and fix our threshold value. Now let us define some more terms such as TPR, FPR, TNR, FNR, Precision, and Accuracy. Positive (P) : Observation is positive (for example: Diabetic).

Negative (N): Observation is not positive (for example: Not Diabetic).

True Positive (TP): Observation is positive, and is predicted to be positive.

False Negative (FN): Observation is positive, but is predicted negative.

True Negative (TN): Observation is negative, and is predicted to be negative.

False Positive (FP): Observation is negative, but is predicted positive.

Total = TP+TN+FP+FN

Accuracy = (TP+TN)/TP+TN+FP+FN

Sensitivity or Recall or TPR = TP/(TP+FN)

Precision = TP/(TP+FP)

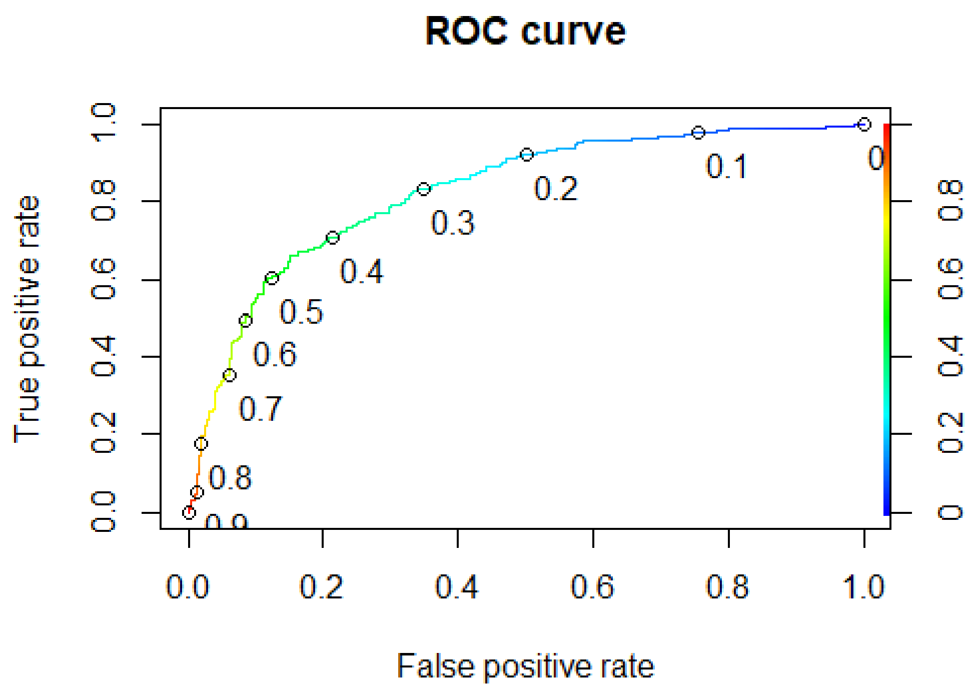
Specificity or TNR = TN/(TN+FP)

F-Measure = (2RecallPrecision)/(Recall+Precision) This is a weighted average of the true positive rate (recall) and precision.

Cohen's Kappa: This is essentially a measure of how well the classifier performed as compared to how well it would have performed simply by chance.

Now here we have a trade-off, the tradeoff is that we can't increase both sensitivity and specificity at the same time, and we need to select one of the two. So in our case of predicting diabetes, it is better to increase TPR and compromise on FPR. This is because this model will be used at the preliminary stage to identify if the patient is diabetic or not. So the error of marking a truly diabetic patient as without diabetes would be bad compared to the error of marking a non-diabetic patient. The non-diabetic patient will go through further screening, and he/she will come to know that they do not have diabetes. So we would like to increase sensitivity for this situation and compromise on specificity.

```r
ROCRpred = prediction(train_set_logistic$Prediction,
train_set_logistic$Outcome)
ROCRperf = performance(ROCRpred, "tpr", "fpr")
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-
0.2,1.7), main="ROC curve")
```



The threshold is varied from 0 to 1 and the ROC curve is plotted for the same. An Ideal ROC curve will look like a box. The more the curve extends to the top left corner, the better the model. An ideal model will have the ROC curve as a square and the area under this curve under ideal situation will be 1. For a null model we will have the ROC curve to be a straight line with slope 1 and passing through the origin. The area under the ROC curve for a null model will be 0.5. Our ROC curve has more AUC than 0.5, as a result it performs better than a null model. We should choose the threshold value from this curve such that the ROC curve saturates and shows not much increase in True Positive Rate for decrease in threshold value. So we will choose 0.30 to be our threshold value for our model.

Confusion Matrix

```
train_set_logistic$Prediction=ifelse(train_set_logistic$Prediction >
0.30,1,0)
table(train_set_logistic$Outcome, train_set_logistic$Prediction)

##
##       0   1
##   0 210 111
##   1  32 159

confusionMatrix(as.factor(as.numeric(train_set_logistic$Prediction)),as.facto
r(train_set_logistic$Outcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 210  32
##          1 111 159
##
##                Accuracy : 0.7207
##                  95% CI : (0.6797, 0.7592)
##     No Information Rate : 0.627
##     P-Value [Acc > NIR] : 4.774e-06
##
##                   Kappa : 0.4491
##
##  Mcnemar's Test P-Value : 6.906e-11
##
##             Sensitivity : 0.6542
##             Specificity : 0.8325
##          Pos Pred Value : 0.8678
##          Neg Pred Value : 0.5889
##              Prevalence : 0.6270
##          Detection Rate : 0.4102
##    Detection Prevalence : 0.4727
##       Balanced Accuracy : 0.7433
##
##        'Positive' Class : 0
##
```

For a threshold of 0.3, the model performs well and gives an accuracy of 75.2% on train data. Both the sensitivity and specificity are also high.

```
test_set_logistic$Prediction=ifelse(test_set_logistic$Prediction > 0.30,1,0)
table(test_set_logistic$Outcome, test_set_logistic$Prediction)

##
##       0   1
##   0 139  40
##   1  18  59

confusionMatrix(as.factor(as.numeric(test_set_logistic$Prediction)),as.factor
(test_set_logistic$Outcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 139  18
##          1  40  59
##
##                Accuracy : 0.7734
##                  95% CI : (0.7172, 0.8232)
##     No Information Rate : 0.6992
##     P-Value [Acc > NIR] : 0.004965
##
```

```
##                   Kappa : 0.5019
##
##  Mcnemar's Test P-Value : 0.005826
##
##             Sensitivity : 0.7765
##             Specificity : 0.7662
##          Pos Pred Value : 0.8854
##          Neg Pred Value : 0.5960
##              Prevalence : 0.6992
##          Detection Rate : 0.5430
##    Detection Prevalence : 0.6133
##       Balanced Accuracy : 0.7714
##
##        'Positive' Class : 0
##
```

The model performs similarly with the test data and has nearly the same accuracy, sensitivity and specificity with that of the train data. This is a good indication that the model is not trying to over fit or under fit the data.

Decision Tree

Let us proceed to develop our next model, Decision Trees Decision trees are non-parametric models meaning that they don't have any underlying assumptions about the distribution of data/errors. Hence we don't have to check for assumption, we can go ahead to develop the model

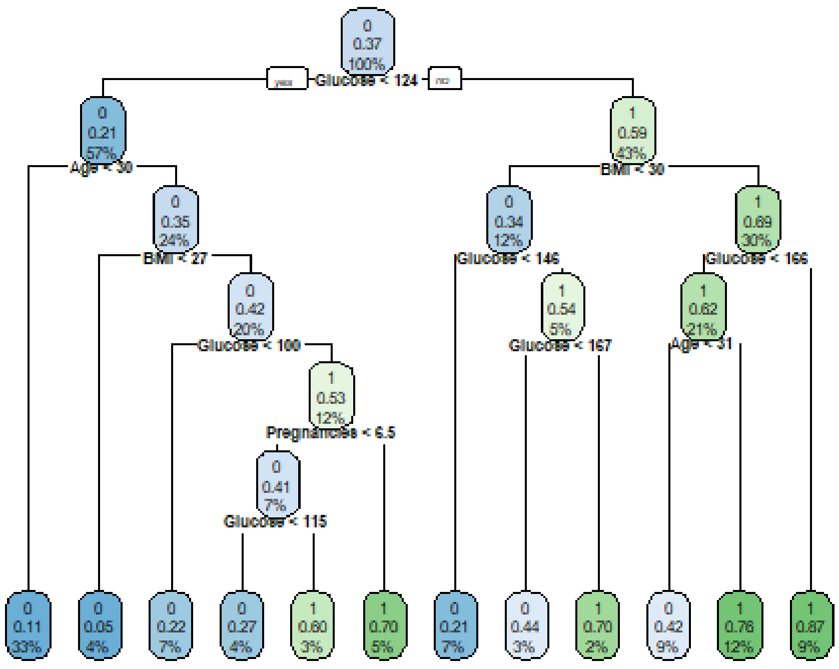
```
train_set_dt=train_set
test_set_dt=test_set
dt_model=rpart(Outcome~., data=train_set_dt, method = 'class')
dt_model_forplot=rpart(Outcome~., data=train_set_dt[,c(1,2,6,8,9)], method =
'class') # for better visualization of the tree.
summary(dt_model)
```

```
## Call:
## rpart(formula = Outcome ~ ., data = train_set_dt, method = "class")
##   n= 512
##
##           CP nsplit rel error    xerror       xstd
## 1 0.20942408      0 1.0000000 1.0000000 0.05729291
## 2 0.10471204      1 0.7905759 0.8481675 0.05509639
## 3 0.02443281      2 0.6858639 0.8376963 0.05491146
## 4 0.01308901      5 0.6125654 0.8115183 0.05442916
## 5 0.01000000     13 0.4973822 0.7696335 0.05359654
##
## Variable importance
##                Glucose                    Age                    BMI
##                     35                     15                     14
##          BloodPressure                Insulin            Pregnancies
##                      9                      9                      8
## DiabetesPedigreeFunction            SkinThickness
##                         5                        5
##
## Node number 1: 512 observations,    complexity param=0.2094241
##   predicted class=0  expected loss=0.3730469  P(node) =1
##     class counts:   321   191
##    probabilities: 0.627 0.373
##   left son=2 (294 obs) right son=3 (218 obs)
##   Primary splits:
##       Glucose     < 123.5  to the left,  improve=36.315480, (0 missing)
##       BMI         < 29.85  to the left,  improve=21.949490, (0 missing)
##       Age         < 28.5   to the left,  improve=20.734060, (0 missing)
##       Pregnancies < 6.5    to the left,  improve=10.641320, (0 missing)
##       Insulin     < 143    to the left,  improve= 9.869195, (0 missing)
##   Surrogate splits:
##       Insulin       < 119.5  to the left,  agree=0.678, adj=0.243, (0
split)
##       Age           < 34.5   to the left,  agree=0.627, adj=0.124, (0
split)
##       BloodPressure < 81     to the left,  agree=0.617, adj=0.101, (0
split)
##       SkinThickness < 32.5   to the left,  agree=0.605, adj=0.073, (0
split)
##       BMI           < 39.75  to the left,  agree=0.604, adj=0.069, (0
split)
##
## Node number 2: 294 observations,    complexity param=0.01308901
##   predicted class=0  expected loss=0.2108844  P(node) =0.5742188
##     class counts:   232    62
##    probabilities: 0.789 0.211
##   left son=4 (171 obs) right son=5 (123 obs)
##   Primary splits:
##       Age                      < 29.5   to the left,  improve=8.137603, (0
missing)
##       Pregnancies              < 6.5    to the left,  improve=7.460597, (0
missing)
##       BMI                      < 26.45  to the left,  improve=7.133960, (0
missing)
##       Glucose                  < 99.5   to the left,  improve=5.730828, (0
missing)
##       DiabetesPedigreeFunction < 0.6345 to the left,  improve=3.766837, (0
missing)
##   Surrogate splits:
##       Pregnancies              < 4.5    to the left,  agree=0.816,
adj=0.561, (0 split)
##       BloodPressure            < 71     to the left,  agree=0.677,
adj=0.228, (0 split)
##       SkinThickness            < 7.5    to the right, agree=0.646,
adj=0.154, (0 split)
##       DiabetesPedigreeFunction < 0.7015 to the left,  agree=0.636,
adj=0.130, (0 split)
##       Insulin                  < 9      to the right, agree=0.619,
adj=0.089, (0 split)
##
## Node number 3: 218 observations,    complexity param=0.104712
##   predicted class=1  expected loss=0.4082569  P(node) =0.4257812
##     class counts:    89   129
##    probabilities: 0.408 0.592
##   left son=6 (64 obs) right son=7 (154 obs)
##   Primary splits:
##       BMI                      < 30.05  to the left,  improve=11.143590,
(0 missing)
##       Glucose                  < 166.5  to the left,  improve= 8.803337,
(0 missing)
##       DiabetesPedigreeFunction < 0.3185 to the left,  improve= 5.899872,
(0 missing)
##       BloodPressure            < 49     to the right, improve= 4.608324,
(0 missing)
##       Age                      < 24.5   to the left,  improve= 4.602747,
(0 missing)
##   Surrogate splits:
##       Age     < 61.5   to the right, agree=0.725, adj=0.062, (0 split)
##       Glucose < 124.5  to the left,  agree=0.711, adj=0.016, (0 split)
##
## Node number 4: 171 observations
##   predicted class=0  expected loss=0.1111111  P(node) =0.3339844
##     class counts:   152    19
##    probabilities: 0.889 0.111
##
## Node number 5: 123 observations,    complexity param=0.01308901
##   predicted class=0  expected loss=0.3495935  P(node) =0.2402344
##     class counts:    80    43
##    probabilities: 0.650 0.350
##   left son=10 (22 obs) right son=11 (101 obs)
##   Primary splits:
##       BMI                      < 26.8   to the left,  improve=4.956562, (0
missing)
##       Insulin                  < 140    to the left,  improve=4.239721, (0
missing)
##       DiabetesPedigreeFunction < 0.202  to the left,  improve=3.963299, (0
missing)
##       Glucose                  < 99.5   to the left,  improve=3.856708, (0
missing)
##       Age                      < 56.5   to the right, improve=2.161533, (0
missing)
##
## Node number 6: 64 observations,    complexity param=0.01308901
##   predicted class=0  expected loss=0.34375  P(node) =0.125
##     class counts:    42    22
##    probabilities: 0.656 0.344
##   left son=12 (38 obs) right son=13 (26 obs)
##   Primary splits:
##       Glucose                  < 145.5  to the left,  improve=3.320344, (0
missing)
##       BMI                      < 23.2   to the left,  improve=2.800926, (0
missing)
##       Age                      < 26.5   to the left,  improve=2.322964, (0
missing)
##       Pregnancies              < 2.5    to the left,  improve=1.866813, (0
missing)
##       DiabetesPedigreeFunction < 0.3185 to the left,  improve=1.672654, (0
missing)
##   Surrogate splits:
##       Pregnancies              < 7.5    to the left,  agree=0.625,
adj=0.077, (0 split)
##       BloodPressure            < 52     to the right, agree=0.625,
adj=0.077, (0 split)
##       BMI                      < 29.1   to the left,  agree=0.625,
adj=0.077, (0 split)
##       DiabetesPedigreeFunction < 0.14   to the right, agree=0.609,
adj=0.038, (0 split)
## 
## Node number 7: 154 observations,    complexity param=0.02443281
##   predicted class=1  expected loss=0.3051948  P(node) =0.3007812
##     class counts:    47   107
##    probabilities: 0.305 0.695 
##   left son=14 (108 obs) right son=15 (46 obs)
##   Primary splits:
##       Glucose                  < 165.5  to the left,  improve=4.006535, (0
missing)
##       BloodPressure            < 56     to the right, improve=3.095861, (0
missing)
##       Age                      < 28.5   to the left,  improve=2.944012, (0
missing)
##       DiabetesPedigreeFunction < 0.429  to the left,  improve=2.575238, (0
missing)
##       SkinThickness            < 6.5    to the right, improve=2.464999, (0
missing)
##   Surrogate splits:
##       DiabetesPedigreeFunction < 1.3925 to the left,  agree=0.727,
adj=0.087, (0 split)
##       SkinThickness            < 59.5   to the left,  agree=0.714,
adj=0.043, (0 split)
##       Insulin                  < 389.5  to the left,  agree=0.708,
adj=0.022, (0 split)
##       Age                      < 56     to the left,  agree=0.708,
adj=0.022, (0 split)
## 
## Node number 10: 22 observations
##   predicted class=0  expected loss=0.04545455  P(node) =0.04296875
##     class counts:    21     1
##    probabilities: 0.955 0.045
##
## Node number 11: 101 observations,    complexity param=0.01308901
##   predicted class=0  expected loss=0.4158416  P(node) =0.1972656
##     class counts:    59    42
##    probabilities: 0.584 0.416
##   left son=22 (37 obs) right son=23 (64 obs)
##   Primary splits:
##       Glucose                  < 99.5   to the left,  improve=4.653766, (0
missing)
##       DiabetesPedigreeFunction < 0.2    to the left,  improve=3.855842, (0
missing)
##       Insulin                  < 140    to the left,  improve=2.756054, (0
missing)
##       BMI                      < 38.95  to the left,  improve=1.670378, (0
missing)
##       SkinThickness            < 37.5   to the right, improve=1.430511, (0
missing)
##   Surrogate splits:
##       DiabetesPedigreeFunction < 0.109  to the left,  agree=0.653,
adj=0.054, (0 split)
##
## Node number 12: 38 observations
##   predicted class=0  expected loss=0.2105263  P(node) =0.07421875
##     class counts:    30     8
##    probabilities: 0.789 0.211
##
## Node number 13: 26 observations,    complexity param=0.01308901
##   predicted class=1  expected loss=0.4615385  P(node) =0.05078125
##     class counts:    12    14
##    probabilities: 0.462 0.538
##   left son=26 (15 obs) right son=27 (11 obs)
##   Primary splits:
##       Insulin       < 14.5   to the left,  improve=1.3594410, (0 missing)
##       Glucose       < 166.5  to the left,  improve=0.8480769, (0 missing)
##       SkinThickness < 16.5   to the left,  improve=0.8480769, (0 missing)
##       Age           < 41     to the left,  improve=0.7326007, (0 missing)
##       BloodPressure < 74.5   to the right, improve=0.6230769, (0 missing)
##   Surrogate splits:
##       SkinThickness            < 7      to the left,  agree=0.962,
adj=0.909, (0 split)
##       BloodPressure            < 74.5   to the right, agree=0.731,
adj=0.364, (0 split)
##       Pregnancies              < 1.5    to the right, agree=0.692,
adj=0.273, (0 split)
##       DiabetesPedigreeFunction < 0.315  to the left,  agree=0.692,
adj=0.273, (0 split)
##       BMI                      < 24.95  to the left,  agree=0.615,
adj=0.091, (0 split)
```

```
##
## Node number 14: 108 observations,    complexity param=0.02443281
##   predicted class=1  expected loss=0.3796296  P(node) =0.2109375
##     class counts:    41    67
##    probabilities: 0.380 0.620
##   left son=28 (45 obs) right son=29 (63 obs)
##   Primary splits:
##       Age                      < 30.5   to the left,  improve=6.057672, (0
missing)
##       SkinThickness            < 6.5    to the right, improve=3.703704, (0
missing)
##       BloodPressure            < 56     to the right, improve=3.530164, (0
missing)
##       Insulin                  < 329    to the right, improve=2.370370, (0
missing)
##       DiabetesPedigreeFunction < 0.429  to the left,  improve=2.240741, (0
missing)
##   Surrogate splits:
##       Pregnancies   < 4.5    to the left,  agree=0.806, adj=0.533, (0
split)
##       BloodPressure < 71     to the left,  agree=0.667, adj=0.200, (0
split)
##       Insulin       < 188    to the right, agree=0.657, adj=0.178, (0
split)
##       Glucose       < 128.5  to the left,  agree=0.611, adj=0.067, (0
split)
##       SkinThickness < 37.5   to the right, agree=0.611, adj=0.067, (0
split)
##
## Node number 15: 46 observations
##   predicted class=1  expected loss=0.1304348  P(node) =0.08984375
##     class counts:     6    40
##    probabilities: 0.130 0.870
##
## Node number 22: 37 observations
##   predicted class=0  expected loss=0.2162162  P(node) =0.07226562
##     class counts:    29     8
##    probabilities: 0.784 0.216
##
## Node number 23: 64 observations,    complexity param=0.01308901
##   predicted class=1  expected loss=0.46875  P(node) =0.125
##     class counts:    30    34
##    probabilities: 0.469 0.531
##   left son=46 (8 obs) right son=47 (56 obs)
##   Primary splits:
##       DiabetesPedigreeFunction < 0.2    to the left,  improve=3.0178570,
(0 missing)
##       Pregnancies              < 6.5    to the left,  improve=2.7779030,
(0 missing)
##       SkinThickness            < 41.5   to the right, improve=2.0002530,
(0 missing)
##       Age                      < 53.5   to the right, improve=0.9476817,
(0 missing)
##       BloodPressure            < 67     to the right, improve=0.8463424,
(0 missing)
##
## Node number 26: 15 observations
##   predicted class=0  expected loss=0.4  P(node) =0.02929688
##     class counts:     9     6
##    probabilities: 0.600 0.400
##
## Node number 27: 11 observations
##   predicted class=1  expected loss=0.2727273  P(node) =0.02148438
##     class counts:     3     8
##    probabilities: 0.273 0.727
##
## Node number 28: 45 observations,    complexity param=0.02443281
##   predicted class=0  expected loss=0.4222222  P(node) =0.08789062
##     class counts:    26    19
##    probabilities: 0.578 0.422
##   left son=56 (36 obs) right son=57 (9 obs)
##   Primary splits:
##       BloodPressure            < 61     to the right, improve=4.900000, (0
missing)
##       Insulin                  < 252.5  to the right, improve=4.011111, (0
missing)
##       DiabetesPedigreeFunction < 0.3195 to the left,  improve=1.757399, (0
missing)
##       SkinThickness            < 20     to the right, improve=1.364209, (0
missing)
##       Glucose                  < 146.5  to the right, improve=1.340171, (0
missing)
##
## Node number 29: 63 observations
##   predicted class=1  expected loss=0.2380952  P(node) =0.1230469
##     class counts:    15    48
##    probabilities: 0.238 0.762
##
## Node number 46: 8 observations
##   predicted class=0  expected loss=0.125  P(node) =0.015625
##     class counts:     7     1
##    probabilities: 0.875 0.125
##
## Node number 47: 56 observations,    complexity param=0.01308901
##   predicted class=1  expected loss=0.4107143  P(node) =0.109375
##     class counts:    23    33
##    probabilities: 0.411 0.589
##   left son=94 (32 obs) right son=95 (24 obs)
##   Primary splits:
##       Pregnancies   < 6.5    to the left,  improve=2.1696430, (0 missing)
##       SkinThickness < 41.5   to the right, improve=1.4744900, (0 missing)
##       BloodPressure < 67     to the right, improve=0.7889610, (0 missing)
##       Glucose       < 116    to the right, improve=0.6878367, (0 missing)
##       Insulin       < 37.5   to the right, improve=0.6696429, (0 missing)
##   Surrogate splits:
##       Glucose       < 101.5  to the right, agree=0.625, adj=0.125, (0
split)
##       SkinThickness < 30.5   to the left,  agree=0.625, adj=0.125, (0
split)
##       BMI           < 31.6   to the right, agree=0.607, adj=0.083, (0
split)
##       BloodPressure < 59     to the right, agree=0.589, adj=0.042, (0
split)
##       Insulin       < 107.5  to the right, agree=0.589, adj=0.042, (0
split)
##
## Node number 56: 36 observations
##   predicted class=0  expected loss=0.3055556  P(node) =0.0703125
##     class counts:    25    11
##    probabilities: 0.694 0.306
##
## Node number 57: 9 observations
##   predicted class=1  expected loss=0.1111111  P(node) =0.01757812
##     class counts:     1     8
##    probabilities: 0.111 0.889
##
## Node number 94: 32 observations,    complexity param=0.01308901
##   predicted class=0  expected loss=0.46875  P(node) =0.0625
##     class counts:    17    15
##    probabilities: 0.531 0.469
##   left son=188 (17 obs) right son=189 (15 obs)
##   Primary splits:
##       DiabetesPedigreeFunction < 0.423  to the right, improve=2.2120100,
(0 missing)
##       Age                      < 34.5   to the left,  improve=1.1358810,
(0 missing)
##       Glucose                  < 120.5  to the left,  improve=1.0803570,
(0 missing)
##       SkinThickness            < 25     to the right, improve=1.0355390,
(0 missing)
##       Insulin                  < 170.5  to the right, improve=0.6003571,
(0 missing)
##   Surrogate splits:
##       SkinThickness < 7.5    to the right, agree=0.688, adj=0.333, (0
split)
##       Insulin       < 85.5   to the right, agree=0.688, adj=0.333, (0
split)
##       Age           < 34.5   to the left,  agree=0.688, adj=0.333, (0
split)
##       BloodPressure < 71     to the left,  agree=0.625, adj=0.200, (0
split)
##       Glucose       < 107.5  to the left,  agree=0.594, adj=0.133, (0
split)
##
## Node number 95: 24 observations
##   predicted class=1  expected loss=0.25  P(node) =0.046875
##     class counts:     6    18
##    probabilities: 0.250 0.750
##
## Node number 188: 17 observations
##   predicted class=0  expected loss=0.2941176  P(node) =0.03320312
##     class counts:    12     5
##    probabilities: 0.706 0.294
##
## Node number 189: 15 observations
##   predicted class=1  expected loss=0.3333333  P(node) =0.02929688
##     class counts:     5    10
##    probabilities: 0.333 0.667
```

```
rpart.plot(dt_model_forplot, cex=0.5)
```

```
## Warning: Bad 'data' field in model 'call' (expected a data.frame or a
matrix).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



We were also able to plot the tree by selecting very limited number of columns for better visualization. We can see the splits at each level and the number of elements considered at each level. This model is very easy to develop and understand.

```
train_set_dt$Prediction <- predict( dt_model, newdata = train_set_dt, type =
"class" )
test_set_dt$Prediction  <- predict( dt_model, newdata = test_set_dt , type =
"class" )
```

Performance of the Model

Now let us analyze the accuracy and other parameters of the decision tree model.

```
table(train_set_dt$Outcome, train_set_dt$Prediction)

##
##       0   1
##   0 285  36
##   1  59 132

confusionMatrix(as.factor((train_set_dt$Prediction)),as.factor(train_set_dt$O
utcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 285  59
##          1  36 132
##
##                Accuracy : 0.8145
##                  95% CI : (0.778, 0.8472)
##     No Information Rate : 0.627
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5934
##
##  Mcnemar's Test P-Value : 0.024
##
##             Sensitivity : 0.8879
##             Specificity : 0.6911
##          Pos Pred Value : 0.8285
##          Neg Pred Value : 0.7857
##              Prevalence : 0.6270
##          Detection Rate : 0.5566
##    Detection Prevalence : 0.6719
##       Balanced Accuracy : 0.7895
##
##        'Positive' Class : 0
##
```

```
table(test_set_dt$Outcome, test_set_dt$Prediction)

##
##       0   1
##   0 158  21
##   1  31  46
```

```
confusionMatrix(as.factor((test_set_dt$Prediction)),as.factor(test_set_dt$Out
come))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 158  31
##          1  21  46
##
##                Accuracy : 0.7969
##                  95% CI : (0.7423, 0.8444)
##     No Information Rate : 0.6992
##     P-Value [Acc > NIR] : 0.0002773
##
##                   Kappa : 0.4985
##
##  Mcnemar's Test P-Value : 0.2120034
##
##             Sensitivity : 0.8827
##             Specificity : 0.5974
##          Pos Pred Value : 0.8360
##          Neg Pred Value : 0.6866
##              Prevalence : 0.6992
##          Detection Rate : 0.6172
##    Detection Prevalence : 0.7383
##       Balanced Accuracy : 0.7400
##
##        'Positive' Class : 0
##
```

From the above result, we observe that the decision tree gives an accuracy of about 90% on train data and 70% on test data. This is because the decision tree has over fit itself to the train data and hence the accuracy drops in test data. Even though the accuracy is less, the decision tree shows a very high sensitivity on test data.

Random Forest

Let us move on to develop the Random Forest model. Random Forest is one such very powerful ensemble machine learning algorithm which works by creating multiple decision trees and then combining the output generated by each of the decision trees. The random forest algorithm works by aggregating the predictions made by multiple decision trees of varying depth. Every decision tree in the forest is trained on a subset of the dataset called the bootstrapped dataset.

```
train_set_rf=train_set
test_set_rf=test_set
rf_model=randomForest(Outcome~., data=train_set_rf, ntree = 300, mtry = 6,
importance = TRUE, OOB  = TRUE)
print(rf_model)

##
## Call:
##  randomForest(formula = Outcome ~ ., data = train_set_rf, ntree = 300,
mtry = 6, importance = TRUE, OOB = TRUE)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 26.95%
## Confusion matrix:
##     0   1 class.error
## 0 257  64   0.1993769
## 1  74 117   0.3874346
```

The portion of samples that were left out during the construction of each decision tree in the forest are referred to as the Out-Of-Bag (OOB) dataset.

Let us predict the results for train and test data using the random forest model

```
train_set_rf$Prediction <- predict( rf_model, newdata = train_set_rf, type =
"class" )
test_set_rf$Prediction  <- predict( rf_model, newdata = test_set_rf , type =
"class" )
```

Let us calculate the various parameters of the confusion matrix

```
table(train_set_rf$Outcome, train_set_rf$Prediction)

##
##       0   1
##   0 321   0
##   1   0 191

confusionMatrix(as.factor((train_set_rf$Prediction)),as.factor(train_set_rf$O
utcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 321   0
##          1   0 191
##
```

```
##                Accuracy : 1
##                  95% CI : (0.9928, 1)
##     No Information Rate : 0.627
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.627
##          Detection Rate : 0.627
##    Detection Prevalence : 0.627
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
table(test_set_rf$Outcome, test_set_rf$Prediction)
```

```
##
##       0   1
##   0 156  23
##   1  30  47
```

```
confusionMatrix(as.factor((test_set_rf$Prediction)),as.factor(test_set_rf$Out
come))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 156  30
##          1  23  47
##
##                Accuracy : 0.793
##                  95% CI : (0.7381, 0.8409)
##     No Information Rate : 0.6992
##     P-Value [Acc > NIR] : 0.0004738
##
##                   Kappa : 0.4947
##
##  Mcnemar's Test P-Value : 0.4098467
##
##             Sensitivity : 0.8715
##             Specificity : 0.6104
##          Pos Pred Value : 0.8387
##          Neg Pred Value : 0.6714
##               Prevalence : 0.6992
##           Detection Rate : 0.6094
##     Detection Prevalence : 0.7266
##        Balanced Accuracy : 0.7409
##
##         'Positive' Class : 0
##
```

We observe that the random forest performs really well with the train data set and nearly shows a 100% accuracy on the train data but it failed to perform that well in the test data and gave an accuracy of 77% but it is still better than the accuracies of decision tree and logistic regression models. We can call this model also overfitting as the accuracy in test is not as good as in train data set.

Comparison of different models

All the 3 models have nearly the same accuracy. Of the 3 models, Random Forest performs well gives the highest accuracy for the given data. So we can tune the random forest model to perform well and use it to predict whether a Pima Indian Patient has diabetics or not.

Future Work

We can improve these models by adding extra features and fine tuning them. For example, the logistic regression model can be improved by choosing a better threshold value. This can be done by evaluating a cost function for various threshold values. The decision tree model could have been pruned to avoid overfitting. The hyper parameters of Random forest could also be tuned to increase the overall accuracy of the model. Apart from the 3 model we could have tried other advanced models such as Ada Boosting trees, Gradient Boosting in Random forest, Deep Learning and Neural Network models. We could also improve the training of all models by using k-fold cross validation. One more option to improve the model would be to add more external features that can be related to diabetes in Pima Indians and we could also take a larger sample to improve the quality of our models.

References

1. https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes
2. https://www.endocrineweb.com/conditions/type-1-diabetes/what-insulin
3. https://en.wikipedia.org/wiki/Pima_people
4. https://bmcendocrdisord.biomedcentral.com/articles/10.1186/s12902-019-0436-6
5. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
6. https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
7. https://webfocusinfocenter.informationbuilders.com/wfappent/TLs/TL_rstat/source/LogisticRegression43.htm
8. https://discuss.analyticsvidhya.com/t/what-is-null-and-residual-deviance-in-logistic-regression/2605

9. https://stats.stackexchange.com/questions/110969/using-the-caret-package-is-it-possible-to-obtain-confusion-matrices-for-specific
10. https://www.healthnewsreview.org/toolkit/tips-for-understanding-studies/understanding-medical-tests-sensitivity-specificity-and-positive-predictive-value/
11. https://medium.com/analytics-vidhya/a-guide-to-machine-learning-in-r-for-beginners-part-5-4c00f2366b90
12. https://en.wikipedia.org/wiki/Receiver_operating_characteristic
13. http://ethen8181.github.io/machine-learning/unbalanced/unbalanced.html
14. http://r-statistics.co/Logistic-Regression-With-R.html
15. https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1